40

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/649,270	08/28/2000	Lawrence A. Crowl	SUN1P380/P4501	6759

22434 7590 08/25/2005

BEYER WEAVER & THOMAS LLP
P.O. BOX 70250
OAKLAND, CA 94612-0250

EXAMINER	
VU, TUAN A	
ART UNIT	**PAPER NUMBER**
2193	

DATE MAILED: 08/25/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 10/03)

Office Action Summary	Application No. 09/649,270	Applicant(s) CROWL ET AL.	
	Examiner Tuan A. Vu	Art Unit 2193	

-- *The MAILING DATE of this communication appears on the cover sheet with the correspondence address* --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 18 May 2005.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1,4-10,12-16 and 19-22 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1,4-10,12-16 and 19-22 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

TV

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 1/13/2005.

As indicated in Applicant's response, claims 1, 4-5, 10, 13, 16, 90 and 21 have been amended. Claims 1, 4-10,12-16 and 19-22 are pending in the office action.

Claim Rejections - 35 USC § 112

2. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

3. Claims 1, 10, 16, and 21 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention. The limitation recited as '...defines a context within which the differential name has an unambiguous meaning...' (li. 11, 10, 12, 13 of claims 1, 10, 16, 21, respectively) is not mentioned anywhere in the specifications. The specifications mention about some context-independent name of a container (Specs. pg. 9) and parsing for parts of the encoded name that are different (pg. 11); and distinct substrings with respect to a container (pg. 12); but nowhere is the word 'context' described as being defined in conjunction with 'differential name' having 'an unambiguous meaning'. One skill in the art would not be able to construe what this so-recited 'context' amounts to in order for some 'differential name' to have some unambiguous meaning, particularly when the specifications do not touch on how the 'meaning' of a differential name is

implemented or established mainly in association with anything termed as a 'context', or related to 'unambiguous meaning'.

The limitation would be treated as though the differential name being parsed among others is perceived in a context that each is distinct from one another.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1, 4-8, 10, 12, 16, 19, 21 and 22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Unger et al, USPN: 5,991,713 (hereinafter Unger), in view of Porter et al., USPN: 6,163, 811, (hereinafter Porter); and further in view of Ainon, "Storing text using integer codes", 1986, *Proceedings of the 11th conference on Computational linguistics* (hereinafter Ainon)

As per claim 1, Unger discloses a method of compiling source program into a compiler products (e.g. Fig. 7) in a compressed form, said method comprising:

receiving a source program including one or more program symbols and non-program information (e.g. *vocabulary word, token* – col. 9, lines 5-30; *numeric string, currency symbols* – col. 10, lines 23-39; steps *212-214, 222* – Fig. 8 – Note: non-program symbols are pointer information for tag correspondence);

encoding a program symbol name to produce an encoded program name (see Fig. 8; Note: some form of encoding of textual and numerical symbols is inferred from encoding text

strings of HTTP page/document under HTML format and protocol - e.g. col. 4, lines 25-55 – and the fact of compressing symbol name or text of HTML source file implicitly discloses a form of encoding using a algorithm) , without changing the non-program symbol information (e.g. step 222 – Fig. 8);

producing a compressed compiler product based on at least the compressed compiler related information (e.g. step *218, 219,* Fig. 8; col. 12, lines 1-6).

But Unger does not disclose encoded program symbol of a source program being a high-level programming language. Compressing source program files like by Unger is furthered into compressing of a programming language written in C++, Java, Pascal, or Fortran via Porter. Porter, in a method to compress and transmit application code using tokenized source data and symbol storage and web page for source file (e.g. col. 2, line 21-34) analogous to Unger, discloses applying compression to Java program source code (e.g. col. 4, lines 6-24; Figs. 1-3). It would have been obvious for one of ordinary skill in the art at the time the invention was made to use a source program written in Java as taught by Porter and submit it to the compression process used by Unger because browser and supporting programming language like Java and its products are well-known and using Java code in a distributed scheme (see Porter: Background) is well-adapted for their portability and platform independency as well as support of many browser applications and material, i.e. HTML, XML applications just as suggested in Unger's invention.

Further Unger does not explicitly disclose compressed form containing differential names; determining a differential name for the encoded program symbol relative to a base symbol for the program symbol, the differential name having a reduced-size format as compared

to the encoded program symbol name; nor does Unger disclose that replacing the encoded symbol name with the differential name to facilitate producing the compiler product includes the generated differential name. Based on the teaching by Porter's base/delta formatting (see Porter: col. 6, lines 44-54) for providing compressed files and underlying results understood from the techniques of compression by Unger (e.g. *dictionary ... encoding, run-length encoding* - col. 10, line 23 to col. 11, line 44; *Huffman* -- col. 8. lines 45-52 – Note: all of these compression algorithms aiming at reducing repeating patterns so that only non-repeating elements are stored already imply a form of base/delta compression), the concept of replacing the encoded symbol name with the differential and producing a encoded representation with generating a differential portions of such representation with respect to a base element comprising a source file is strongly suggested or disclosed.

Unger and Porter thus combined and disclosing a context in which the differential name is such that each is distinct from one another with respect to a base container, i.e. base/delta form – have disclosed 'defines a context within which the differential name has an unambiguous meaning', and that the differential name is formed of characters constituting a subset of the encoded program symbol name (Note: every set of characters extracted to be a delta set from the original non-compressed encoded form as by Unger or Porter reads on subsets of a bigger whole of characters)

However, Unger combined with Porter does not explicitly teach that the base symbol is a containing scope being one of: a namespace, a package, a module, a container object or a function. The concept of base/delta by Porter is suggestive that the base part is portion common to a programming language constructs (or language encoded form) to which non-common or

differential parts of the encoded file can be added to for composing a compressed programming language file.

Similar to identifying a common repeated strings (Note: encoded representation of a high level source file amounts to alpha numerical strings) and appending the rest of the sequences to this common group as mentioned above with Unger's run-length encoding among other methods and Porter's base/delta teaching, another technique by Ainon discloses a common ancestor/family group based on a family of word type (see pg. 419-420), which is equivalent to a containing scope or a namespace or a object-oriented package. Based on the understanding that OO programming language objects like C++, Java as taught by Porter necessarily contain a class or a ancestor object from which subclass or objects can be derived as extensions to those common class/package, the teaching by Ainon emphasizes on a container object (or ancestor object) common to some differential/extension elements of a language that can be appended to form the compressed form as suggested by Porter/Unger base/delta scheme. It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the compression by Unger or Porter using a base group, or base symbol as claimed, identified as an ancestor or family of word type, i.e. a container object or a ancestor package, as taught by Ainon, because since Unger or Porter discloses compression based on identifying common and repeated dictionary/symbol structures or language sequences, using the criteria of identifying common atomic elements based on family or ancestor of word type as taught by Ainon would also be a form of alleviating repeats as mentioned above, particularly if the source files to compress are object-oriented language constructs as by Porter as mentioned above, thus allow providing a compressed form founded on a base object and a delta object as set forth above.

As per claim 4, Unger discloses encoding an encoded program symbol name in the compiler information (Fig. 8; col. 8. lines 45-52; col. 4, lines 25-55) with an encoded format but does not explicitly specify identifying an encoded program symbol name that is encoded in an extended format encoding. But the fact of encoding by Unger, i.e. applying additional structure or code to hide the original atomic element of sequence of characters, text or token so to encode a source string or stream implicitly discloses extended format of such original source of data or characters otherwise encoding is no longer encoding because it exposes the very format of the original source data.

As per claims 5 and 6, Unger discloses determining an encoded program symbol name identifier (e.g. *token* – col. 8, line 54 to col. 9, line 14; *token range* – Fig. 9); and attaching such identifier to the encoding (Fig. 8; *token numbers* -- col .9, lines 39-54; steps *210,212* – Fig. 8 – Note: a string of tokens being identifiable by some integer reads on encoded program symbol name identifier). However, Unger does not disclose determining if an augmented differential encoding is needed, attaching a encoded program symbol name identifier to the differential encoding; nor does Unger teach that such identifier is a base symbol name. But these limitations have been addressed for obviousness in claim 1 above (with Ainon and Storer's teachings) and are herein rejected for the same rationale therein because identifying a common group, or a base group identified by some structure, to factor out is equivalent to determining whether a augmented differential encoding is needed and attaching the differential encoding to such common base type identifier and attaching thereto the differential encoding as suggested by Ainon and Storer.

As per claim 7, Unger discloses that the source program to compile is HTML material such as HTML, XML, SGML files (e.g. col. 5, lines 1-12); but does not specify that the source program is a programming language written in C++, Java, Pascal, or Fortran. Porter, in a method to compress application code using tokenized source data and symbol storage and web page for source file (e.g. col. 2, line 21-34) as mentioned above, discloses applying compression to Java program source code (e.g. col. 4, lines 6-24; Figs. 1-3). It would have been obvious for one of ordinary skill in the art at the time the invention was made to use a source program written in Java as taught by Porter and submit it to the compression process used by Unger because Java language programming and its products are well-known for their portability and platform independency as well as support of many browser applications and material, i.e. HTML, XML applications just as suggested in Unger's invention, and for the reasons according to the rationale as set forth in claim 1.

As per claim 8, Unger discloses parsing and compressing browser documents but Porter from above discloses compressing of program code using tokenized process analogous to Unger. In view of the rationale in claim 1 using Porter's teachings for addressing the program code symbols parsing, the limitation as to compress an object code file would also have been obvious herein because one ordinary skill in the art would be motivated to combine using the browser compiler/parsing schemes by Unger and enhance those with capabilities to parse program code and compress such code as taught by Porter in order to yield compressed version of such parsed program as intended by Unger, because object code delivery in compressed form would facilitate distribution and storage resources saving; and for the reasons according to the rationale as set forth in claim 1.

As per claim 10, Unger discloses a method for generating uncompressed symbol names being associated (col. 9, lines 5-30; col. 10, lines 23-39) with compiler information, said method comprising:

receiving a source program including one or more program symbols and non-program information (e.g. .g. *vocabulary word, token* – col. 9, lines 5-30; *numeric string, currency symbols* – col. 10, lines 23-39; steps *212-214, 222* – Fig. 8)

identifying a compressed encoded symbol name being associated with compiler information (*token, words, strings* – col. 16, lines 8-17; Fig. 5 – Note: token is compiler information and symbol formatted inside a HTTP protocol is equivalent to being encoded under such HTML format or HTTP protocol);

obtaining information relating to the compressed symbol name (e.g. *dictionaries* – col. 38-55); and

decompressing the compressed encoded symbol name to obtain an encoded symbol name in a uncompressed form (e.g. col. 15, line 60 to col. 16, line 7).

Unger does not explicitly disclose compiler information containing differential names; program symbol names in a differential format and extracting a differential symbol and a reference to a base symbol and using it to locate a non-differential name. The limitation to encode a differential form of an encoded source file and determining which encoded form is a base symbol has been addressed in claim 1 using the base/delta teaching by Porter combined with the run-length encoding by Unger.

Unger and Porter so combined thus disclosing a context in which the differential name is such that each is distinct from one another with respect to a base container, i.e. base/delta form –

have disclosed 'defines a context within which the differential name has an unambiguous meaning'.

However, Unger does not explicitly teach that the base symbol is a containing scope being one of: a namespace, a package, a module, a container object or a function. The concept of providing compressed source file having containing scope such as a namespace or a container scope of a programming language has been mentioned via Porter's compression of programming language source files in claim 1; and combined with the ancestor container by Ainon this limitation would have been obvious if source files to compress using Unger encoding scheme is enhanced with the base/delta by Porter wherein a portion common to a programming language constructs (or language encoded form) is isolated so that non-common or differential parts of the encoded file can be added to for composing a compressed programming language file (refer to rationale of claim 1).

Nor does the combination of Unger from above explicitly disclose replacing the differential name with the non-differential name from the base symbol to obtain the uncompressed from thereby producing a decompressed compiler product. But this limitation as to reverse the process of compressing would have been obvious in light of the compression techniques as set forth above because one skill in the art would be motivated to provide the use of reference to a base symbol container and differential name as set forth above by Unger/Porter/Ainon in order to reconstruct the uncompressed product as disclosed by Unger in light of the rationale from above.

As per claim 12, Unger discloses Run-Length encoding which implicitly teaches a combination of small variations and a grouping of repeated elements, hence has suggested a

container for grouping the repeated elements in the sequence. Further, Porter teaches base/delta format and Ainon teaches a base type group based on dictionary families of words (see claim 1); hence the combination of Unger in view of Ainon/Porter would render this container limitation obvious in light of the rationale as set forth in claim 1.

As per claim 16, this is a computer-readable medium claim corresponding to claim 1 above, including all the limitations therein, hence is rejected herein for the same reasons as set forth therein.

As per claim 19, this is a computer-readable medium claim corresponding to claim 4 above, including all the limitations therein, hence is rejected herein for the same reasons as set forth therein.

As per claim 21, Unger discloses a computer-readable medium including a computer program encoded program symbol names in a uncompressed from and associated with compiler information, said computer medium comprising the corresponding limitations of claim 10 above. Hence the claim is rejected herein for the same reasons as set forth therein.

As per claim 22, this claim includes the base program symbol being a container and corresponds to claim 12; hence is rejected with the rejection as set forth therein.

6. Claim 13 is rejected under 35 U.S.C. 103(a) as being unpatentable over Unger et al, USPN: 5,991,713 (hereinafter Unger), in view of Porter et al., USPN: 6,163, 811.

As per claim 13, Unger discloses a compiler system suitable for compiling source programs, said compilation system comprising:

an enhanced compiler suitable for generation of enhanced compiler products (products *54-62* – Fig. 7), such compiler being operable to compile a source program having at least one

program symbol name to produce the enhanced compiler products, such products having a reduced size in comparison with conventional compiler products (e.g. col. 1, line 47 to col. 2, line 39; steps *210, 212, 213, 218* -- Fig. 8); and

at least one enhanced non-compiler component operable to understand and utilize the enhanced compiler products (e.g. *proxy* – col. 14, lines 14-58 – Note: decompressing compressed data is equivalent to parsing and making use of compressed compiler products).

But Unger does not explicitly specify that the source program has at least one compressed encoded program symbol name, nor does Unger explicitly disclose including one or more differential names corresponding to a program symbol names. But this encoded symbol with differential symbols of a source file being formed with a base symbol has been addressed using the programming language compression by Porter teaching a base/delta in conjunction with suggestion by Unger in Run-Length encoding in claim 1.

7. Claim 14 is rejected under 35 U.S.C. 103(a) as being unpatentable over Unger et al, USPN: 5,991,713 (hereinafter Unger), in view of Porter et al., USPN: 6,163, 811, (hereinafter Porter); and further in view of Ainon, "Storing text using integer codes", 1986, *Proceedings of the 11th conference on Computational linguistics.*

As per claim 14, with respect to claim 13, Unger discloses using encoding technique to reduce size of the enhanced compiler product (Unger: *Huffman* -- col. 8. lines 45-52; *run-length* – col. 11, lines 38-44); but does not specify that such reduction is up to 40 percent of sizes of conventional compiler products. Ainon, in a analogous method to compress data based on dictionary of words using a family type to form a base group as disclosed in claim 1, discloses a better ratio result up to 40% to conventional methods (Table 2 pg. 420 - *Two-byte-word 2.0* and

Huffman 1.9). It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the compressing and encoding by Ainon to the Huffman's technique by Unger because targeting and achieving up to 40% in size reduction would better preserve storage resources as intended in Unger's compression technique.

8. Claims 9 and 20 are rejected under 35 U.S.C. 103(a) as being unpatentable over Unger et al, USPN: 5,991,713, in view of Porter et al., USPN: 6,163, 811; and Ainon, "Storing text using integer codes", 1986, *Proceedings of the 11th conference on Computational linguistics*; as applied to claims 1, 16 and further in view of (no author) G06F011/28 by Derwent 1998-236084, JP Pub N: JP 10074152A (hereinafter JP-DW-1998).

As per claim 9, Unger (with Ainon and Porter teachings) teaches an enhanced compressed compiler product, but fails to specify including therein an object file, an executable or a debugging information. Further, Porter discloses applying compression to Java program source code (e.g. col. 4, lines 6-24; Figs. 1-3), hence has suggested a program object code as mentioned in claim 8. Further, JP-DW-1998, in a debug system for compressing code as mentioned in claim 8 above, discloses compressing both the debug information and executable code in the deliverable that is to be loaded on the target computer (JP-DW-1998: see front page and ABSTRACT). In view of the rationale in claim 8 to combine Unger teachings with Porter's for providing object file code, it would also be obvious for one of ordinary skill in the art at the time the invention was made to further include executable code and debug information in the compressed product as taught by JP-DW-1998 in order to enhance the utilization of the compressed code delivered as suggested by Porter (in combination with Unger) as to facilitate the debugging and additional memory usage as suggested by JP-DW-1998.

As per claim 20, this claim includes the same limitations as claim 15 above, hence is rejected herein for the same reasons as set forth therein.

9. Claim 15 is rejected under 35 U.S.C. 103(a) as being unpatentable over Unger et al, USPN: 5,991,713, in view of Porter et al., USPN: 6,163, 811; as applied to claim 13; and further in view of (no author) G06F011/28 by Derwent 1998-236084, JP Pub N: JP 10074152A.

As per claim 15, this claim includes the same limitations as claims 8 and 9 above, hence is rejected herein for the same reasons as set forth therein.

Response to Arguments

10. Applicant's arguments filed 5/18/2005 have been fully considered but they are not persuasive. Following are the Examiner's pertinent remarks.

Rejections 35 USC §103(a):

(A) Applicant has submitted that Unger teaches comparing '...words to a predetermined dictionary ... are merely characters ... text file being compressed' and thus can not be equated to 'encoded program symbols' of the invention (Appl. Rmrks, pg. 8, 2nd para). The claim specifies an encoded form of a source program being compressed; and the rejection using Unger mentions about encoding of source files the likes of which implicate Huffman encoding or Run-Length encoding. Since it is the encoded format of source files that is studied (from the claim: this format being *sequence of characters*) so as to detect what is a common pattern or base symbol in order for a differential encoding to be created based on such determination, the object of such determination (i.e. base container or differential parts) clearly points to a sequence of alpha-numerical atomic elements, such elements being the result of encoding, i.e. not the original raw text elements (e.g. encoded and being in *characters* form). When Unger's tokens from

parsing a source code are analyzed, or when browser files are packed using the above encoding scheme, those same alpha-numeric sequences would be analyzed and appropriate determination based upon repeated patterns of those atomic elements are effected according to a compression algorithm of choice (Huffman or Run-Length). The claims recite encoded forms, even encoded program symbols, being compressed; and since encoded form implies some alpha-numerical representation as recognized via the parsing or token representation or pre-compression encoding as taught by Unger, the compression by Unger necessarily applies to this form of encoded format of strings albeit their being in character form, i.e. after the textual original dictionary words are encoded (see Unger: col. 9, lines 46-50), hence not the original textual content. The claim does not make it very explicit as to how the so-called 'encoded program symbol' amounts to; and it is the burden of the Applicant to point out from the language of the claim specifics as to how this encoding (of program symbols as recited) is implemented in order to negate Unger's technique to transform the textual symbols so that only after applying some encoding (from the parser) or compression technique (from Huffman encoding) a compressed form is generated, which is shown in the rejection. Since the reference by Unger shows encoding of symbols using some technique of compression and that the encoded form of these symbols are not explicitly disclosed as being absolutely text symbols, it would be an assumption without evidence to say the Unger does not disclose encoded symbols being other than textual symbols. Besides, the claims clearly recite that the encoded program symbols are characters; and characters are known to be the atomic elements of many form of texts, or binary stream or program code; i.e. accusing that Unger's encoded form is just text would contradict with the claim reciting that the very encoded form is character form. Therefore, the arguments are mere assertions and are deemed non

conclusive. The argument further raises the issue that Unger is only purported to encode text file as much as possible while the invention focuses on encoding program symbol names and non-program symbol names. The rejection has shown that some non-symbol information has been also compressed as part of the deliverables (re claim 1) and the rationale to provide why program symbols as Porter's programming language in addition to Unger's simple alpha numeric atomic elements can be compressed has been set forth for that purpose of rendering program symbol encoding obvious. The rationale of the rejection has to be perceived as a combination not as a single reference standpoint. The motivation as to send a encoded and compressed program or application files as taught by Unger so that those compressed files are Java encoded files as by Porter would have been obvious in view of the rationale as set forth in the rejection. In response to applicant's arguments against the references individually, one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.*, 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986).

(B) Applicants have submitted that the 'containing scope' has been amended to make it clear as to specifying the differential name limitation within a context being such that the differential name has a unambiguous meaning (Appl. Rmrks, pg. 8, bottom) and that such differential is now formed of subset of characters of the encoded program (Appl Rmrks, pg. 9, top). First, forming a differential being a subset from a ensemble of characters is not different from the compression by Huffman or Run-Length by Unger or base/delta encoding by Porter; because basically a common scenario is construed: alpha-numerical flow of atomic elements such as characters (as claimed) are analyzed in either Unger's or Porter's method in their encoding scheme so that only

the repeating characters (base container) are extracted such that the distinct (delta part) character sets are appended to those repeating patterns in order to form the final compressed product. Besides, the 'unambiguous meaning' and the 'context' limitations are treated as though they amount to a mere context of getting only distinct character subsets; and this has been addressed in the 112, 1st paragraph as set forth above. The 103(a) type rejection has shown that the limitation about the context thus interpreted has been disclosed. The rejection has also shown why the differential name being a subset of encoded program symbol name would also be what Porter and Unger have done in the encoding of their symbols respectively; hence the subset limitation have been also disclosed in light of the rationale of the rejection combining Unger and Porter. Applicant's claiming that the amended limitations are now clear does not amount to the fact that they can still read on what has been understood/interpreted by one skill in the art --inter alia, the interpretation from the 112 1st paragraph rejection-- in the light of the teachings from the reference; and as it stands, the references still provide 'distinct' subsets of characters for encoding the differential program symbols to produce compressed output.

(C) Applicant has submitted that Unger, Porter or Ainon disclose differential names formed by sequences of characters constituting subsets of encoded program symbol names 'whose exact meaning can be deduced with reference to the containing scope' (Appl. Rmrks, pg. 9, 2nd para)

The differential form is a form interpreted as being detected as different from the common pattern as mentioned above; hence when it is encoded and being incorporated as a differential element of the final compressed form, this differential form is reduced in size. And from Unger to Porter, to Ainon, this form of reducing the size by providing a differential form being thus reduced by appending it to a common container is disclosed because as soon as a differential

form is determined based on the base container, the end result is a reduced form. The language of the claim for specifying further what a containing scope and a differential name respectively amount to apparently does not seem to provide substantial teaching that would enable one skill in the art to perceive noticeable difference (from the prior art) or clear construction of the subject matter (lack of description in the specifications – re 112, 1st paragraph rejection). Hence, as interpreted from one skill of the art when presented with such unclear teaching which is not supported by the specifications, the subject matter is merely subjected to a broad reasonable interpretation; and the result of which is set forth in the rejection. Unger combined with Porter has disclosed a context in which encoded program symbols are extracted as distinct subsets with respect to a common pattern disclosed as a parent/base container. The portion from the specifications cited in pg. 9 of the argument does not clarify the subject matter as claimed which is rejected in the 112, 1st rejection section; hence would not render Ainon or Unger or Porter 's teaching improper in view of the combination used in the rejection because the arguments cannot attack on one reference taken individually. For example, when it comes to addressing the encoded program symbols, the teachings by Ainon cannot be viewed only in regard to his teachings on a two-byte code but also in the sense that Ainon endeavors a reference to a container object, such endeavor enhancing the encoded subsets of differential parts that have already been addressed by Porter or Unger.

(D) Applicant has submitted that Unger only compresses using conventional techniques leaving program symbol names uncompressed; that Porter does not encode program symbol names but merely teaches substituting with tokens; that Ainon adds nothing (Appl. Rmrks, pg. 10, top). From above, section A has mentioned that the limitation recited as 'encoded program

symbol names' does not enforce a particular encoding scheme that clearly distinguishes from the cited parts of Unger. The rejection has mentioned various ways to see that Unger stream of characters or atomic element or tokens are being encoded from the compiler, and from the various techniques of compression. The limitation as recited bears on programming language symbol names until the claim specifies about the nature of the 'containing scope' and Porter has been used to show that a source file thus being compressed by Unger can be a programming language code such as Java, C++, Ada with bearing on using a base and a delta type of encoding. Ainon is reinforcing further the technique of using a container and a reference to such container which would facilitate compression size and decompressing. The apparent issue that none of the above references shows encoding of program symbol names would have to be viewed in light of how the so-recited limitation has been formulated. The phrase 'encoded program symbol names' reads on many elements taught by the references, the likes of which being elements in stream of source files being parsed by Unger's compiler, by Porter's compiler and by Ainon's system of segregating container elements and other non-container elements into specific structures. Encoding merely means that a original sequence of original atomic elements has been changed by some processing unit into a form so as to facilitate its semantic/syntactic management and/or parsing/compiling according to some scheme/rule needed to yield some results for the analysis and possibly format/size control of the stream. Program symbols does not necessarily enforce a program language such as C++ but can be an instruction type of construct inside a script in a HTML web pages. Encoding source file stream into program symbols happens in every compiler-based parser or translation units. Until the claim clearly describes what constitutes such encoded form of program symbols, the limitation as recited would be interpreted as above

and Unger's source file alone has fulfilled one aspect of such 'encoded program symbol name' (refer to claim 1). Nevertheless, the rejection is construed as a combination of teachings rendering obvious the claim as a whole. In response to applicant's arguments against the references individually, one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.,* 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986).

In short, the arguments are not persuasive and the rejections stand.

Conclusion

11. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (272) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (571)272-3719.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence – please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

VAT
August 10, 2005

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100